Práctica 1:

PROGRAMACIÓN CON ASERTOS

OBJETIVO

- Aprender a programar y compilar en Linux.
- Generación de proyectos con *Make*.
- Uso de asertos en C++.

MATERIAL A ENTREGAR

- complejo.h
- complejo.cpp
- complejoMain.cpp
- makefile
- doxigen.cfg y documentación en html
- Ejercicios apartado 1. (Por Aula Virtual)

DESARROLLO

1. Ejercicios previos

Doxygen: Estudiar cómo se deben escribir los comentarios en doxygen. Escribir el comentario para la siguiente función, que devuelve la suma de **a** más **b**:

```
int suma(int a, int b)
```

Make: Estudiar el uso del make. Escribir el fichero makefile adecuado para compilar la presente práctica.

2. Clase Complejo

Se deberá crear una clase complejo que permita manejar números complejos en coordenadas polares. En esta notación, un número complejo se representa mediante su módulo y su argumento (la relación entre módulo y argumento y parte real e imaginaria se puede ver en la implementación parcial de la clase). El argumento lo representaremos siempre en radianes y estará comprendido entre $-\pi$ y $+\pi$.

Al implementar esta clase se debe incluir también un **invariante de clase**. El invariante de clase es un aserto que se ha de cumplir siempre para cada objeto de la clase, si no se cumple significa que los valores del objeto son incorrectos. Normalmente se comprueba al principio y al final de todos los métodos de la clase que pueden modificar el objeto (métodos no **const**), aunque en esta práctica sólo lo pondremos al final de los métodos que modifican el objeto, incluidos los constructores.

Únicamente se deben implementar los métodos presentes en el fichero de cabecera. El método para imprimir un complejo en pantalla (Print()) se puede sustituir por la sobrecarga del operador <<.

3. Programa principal

En el programa principal se deberá probar en primer lugar que la suma y multiplicación funcionan correctamente, sumando y multiplicando 2 números complejos.

A continuación se deberá ejecutar la función genVector con los parámetros $c = (2,\pi/3)$ y $n = 10$ y por último mostrar el vector resultante. Deberá usarse la clase **vector de la STL**. La especificación de la función genVector es la siguiente:

Func genVector(c: Complejo; n: Nat) dev v: Vector[0..n-1] de Complejo

{Pre: cierto}

{Post: $v[0] = (1,0) \wedge \forall\alpha: 1 \leq \alpha \leq n-1: v[\alpha] = v[\alpha-1] * c$ }

Nota: El *main* no deberá contener ningún menú. Todas las operaciones se ejecutarán seguidas, una detrás de otra, sin que se realice ninguna interacción con el usuario.

4. Generación de documentación

Una vez acabado el programa, se deberá ejecutar el Doxygen para generar toda la documentación del programa. Dicha documentación deberá ser generada en html, y deberá ser entregada junto con la práctica.